AFRL-RY-WP-TR-2009-1146

EXPLORING FIELD-PROGRAMMABLE GATE ARRAY (FPGA)-BASED EMULATION TECHNOLOGIES FOR ACCELERATING COMPUTER ARCHITECTURE DEVELOPMENT AND EVALUATION

Chen Chang and Kevin Camera

BEEcube Inc.

APRIL 2009
Final Report

Approved for public release; distribution unlimited.

See additional restrictions described on inside pages

STINFO COPY

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Defense Advanced Research Projects Agency (DARPA) and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RY-WP-TR-2009-1146 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

*//Signature//

KERRY L. HILL
Project Engineer
Advanced Sensor Components Branch
Aerospace Components & Subsystems
Technology Division

//Signature//

BRADLEY J. PAUL, Chief
Chief, Advanced Sensor Components Branch
Aerospace Components & Subsystems
Technology Division
Sensors Directorate

//Signature//

TODD A. KASTLE
Chief, Aerospace Components & Subsystems
Technology Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show "//Signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
April 2009	Final	16 October 2008 – 27 May 2009

4. TITLE AND SUBTITLE
EXPLORING FIELD-PROGRAMMABLE GATE ARRAY (FPGA)-BASED EMULATION TECHNOLOGIES FOR ACCELERATING COMPUTER ARCHITECTURE DEVELOPMENT AND EVALUATION

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8650-09-C-7911

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)
Chen Chang and Kevin Camera

5d. PROJECT NUMBER
ARPR

5e. TASK NUMBER
YD

5f. WORK UNIT NUMBER
ARPRYD0M

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BEEcube Inc.
39465 Paseo Padre Parkway, Suite 3700
Fremont, CA 94538

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

Defense Advanced Research Projects Agency/
Information Processing Techniques Office
(DARPA/IPTO)
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYDI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2009-1146

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
PAO Case Number: DARPA 14349; Clearance Date: 06 Oct 2009. This report contains color.

14. ABSTRACT
This study compares the commercially available emulation hardware platforms with respect to user productivity, accuracy in multi-core processor architectures, and emulation performance. We also study the emerging FPGA-based emulation acceleration strategies and their effect on target system capacity, simulation performance, and the resource requirements for achieving an acceptable application software runtime. This report evaluates the hardware, software, and user support resource requirements for achieving the optimal balance in user design productivity and model accuracy of the target design, and makes recommendations for emulation strategies and platforms for future consumer and military multi-core processor architecture designers.

15. SUBJECT TERMS
multi-core, processor, architectures, emulation, acceleration

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kerry L. Hill
Unclassified	Unclassified	Unclassified	SAR	36	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Table of Contents

List of Figures

List of Tables

Acknowledgements

The study is sponsored by the Defense Advanced Research Projects Agency (DARPA) Information Processing Techniques Office and US Air Force Research Laboratory Sensors Directorate under contract number FA8650-09-C-7911.

The authors would like to thank the following individuals for collaboration with BEEcube on various technical aspects of the study:

- Krste Asanovic, John Wawrzynek, and Daniel Burke at University of California, Berkeley, for their contribution of multi-core processor modeling techniques and classifications.
- Paul Hartke at Xilinx Inc, Durgam Vahia, Gopal Reddy, and Thomas Thatcher at Sun Microsystems Inc. for collaborating on the OpenSPARC implementation for BEE3 systems.

1 Summary

Future multi-core computer processor architecture exploration and development has presented a significant challenge to the research and development community for both consumer and military applications. The traditional software simulation-based evaluation methodology has become extremely slow to the point where it is impractical to perform meaningful evaluation of high-level design decisions for purposes of optimizing system-level energy consumption, cost, and performance. With software simulation, a comprehensive design space exploration is impossible, particularly because it is currently impractical to run full operating systems or application software on these simulation models. However, it is widely understood that hardware/software co-design is a necessary step in optimization of future multi-core architectures. A new generation of Field Programmable Gate Array (FPGA) based emulation acceleration technology for processor design has emerged. It has the capability of running full application software at an emulated speed close to the final target speed of the processor, with the flexibility to easily try out different architectural variations, and the capacity to highly instrument the emulation models.

This study compares the commercially available emulation hardware platforms with respect to user productivity, accuracy in emulating multi-core processor architectures, and emulation performance. We also study the emerging FPGA-based emulation acceleration strategies and their effect on target system capacity, simulation performance, and the resource requirements for achieving an acceptable runtime for software applications. This report evaluates the hardware, software, and user support resource requirements for achieving the optimal balance in user design productivity and model accuracy of the target design, and makes recommendations for emulation strategies and platforms for future consumer and military multi-core processor architecture designers.

2 Introduction

Throughout this report we use the term “target” to describe the system being modeled and “host” to describe the system that runs the model.

There are five general host-level approaches for pursuing parallel computer systems research at scales of 1,000 processor cores and beyond, including a conventional shared-memory multiprocessor (SMP), a cluster of workstations or PCs, a software simulator, custom chip design, and hardware-based emulation.

Cost rules out a large SMP for most researchers. Also, although an SMP can be used to run multiprocessor applications natively, this does not allow experimentation with different architectural configurations or the addition of new features. The hardware architecture is inflexible, which implies that attempts to experiment with different architecture mechanisms through software would result in very poor emulation performance. Furthermore, when executing natively on real hardware it is difficult to observe, reproduce, and measure low-level system interactions. A large microprocessor cluster is a cheaper way of emulating multiple processor cores and can be used to model a multiprocessor system, but is cumbersome and costly to manage and provides only limited observability, reproducibility, and configurability. In addition, the fine-grained synchronization typical of efficient processor and network emulation is too slow on clusters to permit efficient emulation.

The most practical alternative to date has been software simulation, and indeed that has been the vehicle of choice for most architecture research in the last decade. However, for multiprocessor target systems, simulation speed in total instructions per second drops as more target cores are simulated and as operating system effects are included. Further, when detailed timing and power models are added, software simulation slows dramatically. Sampling techniques and parallelization across independent runs can help improve performance for architecture work, but for software development, where performance is limited by the latency to complete a single run with a new version of the application code, these techniques do not help. Therefore software developers rarely use software simulators.

A few groups have completed custom chip prototypes of their proposed architectural ideas. This is an expensive option that experience has shown takes around five years to complete a working prototype of a given set of architectural mechanisms. Although this provides a highly credible proof of concept, there is usually limited ability to experiment with the choices made (although most research prototypes typically include more support for experimentation than a commercial design). Also, the design budget limitations usually result in a prototype that is considerably slower than a contemporary commercial product.

Architecture-level hardware emulation is a compromise among these alternatives. It is much cheaper than custom hardware that small groups can afford highly scalable systems; it is as flexible as software simulators so that designers can rapidly evolve their system hardware architecture; and it is so much faster than simulators that software developers can actually try out new hardware ideas. The credibility of hardware emulation can be much higher than for software simulation, both because entire applications can be run under complete software stacks and also because the model construction is more similar to Integrated Circuit (IC) design and can actually reuse hardware designs as part of a model. In particular, modern FPGA based hardware emulation can achieve clock rates only 10 times slower or less than custom IC prototypes. This relatively high speed is mainly due to the fact that off-the-shelf FPGAs use IC process technology that is two or three generations ahead of that available for research prototypes.

This study explores the different strategies of FPGA-based emulation acceleration, target system capacity, simulation speed, and resource requirements for achieving an acceptable application software run-time. This study compares multiple commercially available emulation hardware host platforms for their productivity and accuracy in modeling processor architecture, power, and timing. This study also evaluates the hardware, software, and user support resource requirements for achieving the optimal balance in user design productivity and model accuracy of the target processor system.

3 Methods, Assumptions, and Procedures

In this section we classify the currently available host emulation platforms. Among all the commercially available hardware emulation systems, the following five uniquely represent the landscape of all state-of-art technology for large scale multi-processor architecture research applications: 1) ASIC (Application Specific Integrated Circuits) gate-level accelerator, 2) Multi-FPGA ASIC emulator, 3) FPGA accelerator card, 4) Front Side Bus (FSB)-FPGA module, and 5) FPGA compute cluster.

3.1 ASIC gate-level accelerator

Figure 1: Cadence Palladium Systems

ASIC gate-level accelerators are specialized emulation systems that target acceleration of ASIC gate-level netlists. A prominent example is the Cadence Palladium (formerly Quickturn) series of products [1]. Typically a flat ASIC gate-level netlist is fed to a large array of custom designed processors tailored for single bit logic operations, each processor executing a large number of software contexts. The outputs of these system are simulation results, much like in software HDL simulators, but at a speed close to a couple of MHz clock rate. Although the simulation speed is orders of magnitude higher than software Register Transfer Level (RTL) simulation, the performance is still a couple of orders magnitude lower than real ASIC clock rates of a few hundreds of MHz. The latest generation of these large emulators can scale emulation logic capacity from just a few million ASIC gates to 256 million gates. However, the cost of these systems is extremely high, in the range of a few million dollars. Hence these systems are typically restricted to a very few elite companies, dedicated for final stage gate level verification of an ASIC gate-level design. These systems are typically built as an integrated unit with a fixed maximum capacity, which is not user friendly for future upgrades or capacity scaling. Since such a system is constructed with a custom processor designed for gate-level verification, the

design inputs are also constrained to gate-level netlists only. Although gate-level netlists can provide the most accurate emulation results with respect to the final silicon implementation, to produce the required netlist format, users are required not only to have a full RTL implementation of the design, but also correctly synthesized gate-level results. With RTL design and ASIC synthesis in the critical path of design explorations, which can take days to weeks, this approach severely limits the scope of design parameter explorations, and hence is not suited for architecture exploration methods at higher abstraction levels.

3.2 Multi-FPGA ASIC emulator

Figure 2: Synopsys HAPS FPGA Emulator System

A multi-FPGA ASIC emulator is a more cost effective approach to the traditional specialized gate-level emulation systems above. The Synplicity HAPS-54 system (now part of Synopsys) [2] is a good example of these FPGA based ASIC emulation boards. Their prominent characteristic is a single PCB (Printed Circuit Board) with large number of the highest available capacity FPGA parts and many digital expansion connectors for custom add-on modules. Costing hundreds of thousands of dollars, these systems are a fraction of the cost of their high-end counter parts, but can provide up to 64 million ASIC gate capacities. Despite the fact that some of these systems can stack individual boards to provide higher capacity, the scalability of such a configuration is limited by the poor signal integrity of stacking multiple expansion I/O connectors vertically. Typically no more than 4 boards can be stacked, providing a maximum of 16 large FPGAs in a system. Input to these systems can range from RTL to gate-level netlists, but typically require the full implementation to be specified to at least the level of synthesizable RTL. Circuit emulation on a single FPGA can achieve up to a couple hundreds of MHz, but over the full system and across multiple stacked boards, the simulation clock rate is typically reduced to 50 MHz or below.

One key difference between FPGA based ASIC emulators and specialized ASIC gate-level accelerators is how each type of emulator handles the ASIC netlist partitioning problem. The specialized gate-level acceleration processors are usually connected to each other on full crossbar interconnects, where each processor can directly connect to any other processors only through communication nodes. Each processor provides a large number of statically scheduled time-multiplexed processing threads, where individual gate logic can be mapped to each of the processing threads. With 128 processing threads or more per processor and full crossbar interconnects, the partitioning problem for specialized gate-level emulators is simply to schedule groups of logic cores onto one or more processors, and dependent logic operations are mapped to subsequent processing threads.

FPGA based emulators, on the other hand, face a much more difficult problem, as with ASIC gate-level accelerators it is necessary to partition arbitrarily large gate-level netlists. Usually the original gate-level netlist is first technology-retargeted to FPGA logic elements, and then partitioned across multiple FPGA chips spatially to achieve the overall required emulation capacity. Since each individual or small number of ASIC logic gates are directly mapped to one or more FPGA logic elements, the entire FPGA array has to run in synchrony in order to faithfully reproduce the original intended ASIC gate-level functionality. The routing resources inside each FPGA are typically a couple orders of magnitude more abundant than inter-FPGA connections, hence the demand of using the highest capacity FPGAs available, and packing as many FPGAs as reasonable PCB technology can yield. This partitioning challenge is the key reason why some commercial FPGA emulator vendors attempt to integrate 16 or more of the largest FPGA chips on the same table-top size PCBs as thick as ¼ inch. Not only are these systems difficult to manufacture, but also the run-time reliability of these large number of FPGAs and dense interconnects routinely turns the end-user debugging of the emulated design into a witch hunt of whether the issue is in the user design or the underlying emulation system. Nevertheless, when designed properly with limited FPGA integration on each PCB, FPGA based emulation system can provide more than a couple of orders of magnitude speedup over specialized gate-level emulators, all at a fraction of the cost.

3.3 FPGA accelerator cards

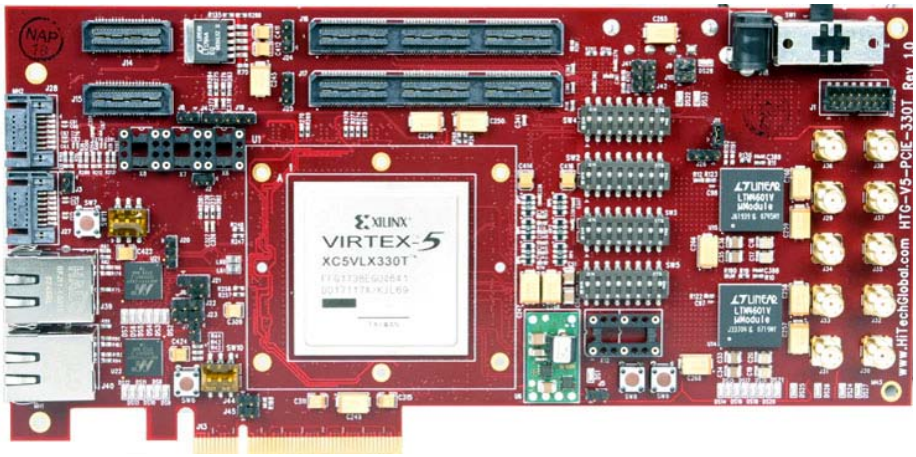

Figure 3: HiTech Global HTG-V5-PCIE 8-lane PCI Express card

FPGA accelerator cards are aimed at design problems that can fit into a single or a couple of large FPGA chips. They are the most prevalent form factor of FPGA based emulation or computing systems, mainly due to their affordable hardware cost of just a few thousand dollars. Such systems typically consist of a single PCB in the Peripheral Component Interconnect (PCI) or PCI-Express (PCIe) card form factor, carrying one to two large FPGAs, a number of memory chips, and other I/O interface components. For example, the HiTech Global HTG-V5-PCIE 8-lane PCI Express card [3] hosts a single Xilinx Virtex-5 LX110T/FX70T/SX95T FPGA, a single DDR2 SO-DIMM up to 2GB capacity, a number of Ethernet interfaces, and other user programmable I/O interfaces. The small and relatively simple PCB allows the system to be priced at only a few thousand dollars, and the system cost is mostly dominated by the FPGA chip cost.

However, the capacity of the system is limited by the PCI card form factor. Although some vendors pack up to 6 FPGAs on a single over-size PCI card, in practice large PCI cards are rather difficult to physically integrate in the host computer system, especially for rack-mount server chassis with limited internal expansion card spaces. On the other hand, single FPGA PCI cards can often fit into higher density 1U or 2U rack mount computer chassis.

The choice of PCI and more recently PCIe as the main communication channel to the host Central Processing Unit (CPU) benefits from software and hardware infrastructure available to a wide range of PCI based computer peripherals. However, the physical chassis limitation usually constrains the host CPU core to FPGA chip ratio to just unity. In addition, the typical latency between the host CPU and the FPGA on the PCIe bus is around tens of microseconds. For ASIC emulation applications, where the PCIe bus is primarily used to transfer emulation inputs and outputs, the host CPU to FPGA latency is not critical; however for hybrid hardware-in-the-loop simulations, this latency can rapidly become the bottleneck of the entire system operation.

3.4 FPGA coprocessor module

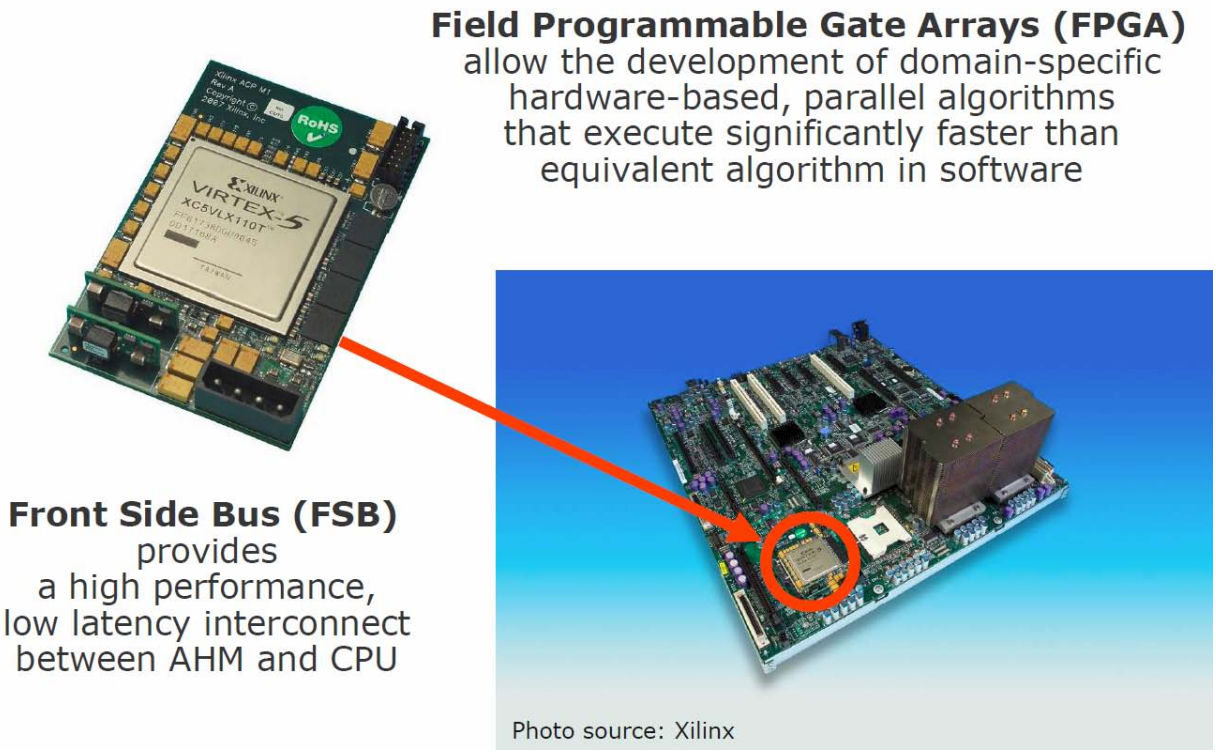

Figure 4: Nallatech Slipstream FSB-FPGA

The FPGA coprocessor module approach is aimed at reducing the host CPU to FPGA latency down to the sub-microsecond range. The basic idea is rather than connecting the FPGA chip as a peripheral on the host CPU PCI/PCIe bus, instead to directly connect it on the main inter-CPU interface. In the case of the Nallatech Slipstream FSB-FPGA [4], a single FSB-base module with a Xilinx Virtex-5 FPGA is connected to the Intel Front Side Bus (FSB) on the host server motherboard. The FSB-base module is dedicated for FSB bridge functionality only, and its FPGA logic cannot be changed by the end-user. Additional FSB-compute and FSB-expansion modules can be stacked on the FSB-base module to expand the FPGA capacity and other I/O interfaces. Each FSB-compute module can carry two FPGAs, and up to two FSB-compute modules can be stacked on top of a single FSB-bridge module. Finally a single FSB-expansion module can be added on the top of the stack, carrying a single FPGA with high speed I/O connector for expansions. A single 4-socket server motherboard can carry up to three FSB-FPGA stacks, with the last socket left for the CPU. Therefore, up to 15 user-accessible FPGAs (3 sockets, each with a stack of two FSB-compute and one FSB-expansion modules) are available in a single 4U host server chassis.

Figure 5: FSB FPGA module stack

The key difference between FPGA coprocessor modules and FPGA accelerator cards is the communication latency from the host computer to each FPGA. Connected directly on the FSB as a coprocessor, the FSB-FPGA approach can share memory coherently with the host CPU and benefits directly from the low host communication latency of hundreds of nanoseconds, rather than several microseconds as in the case of PCI/PCIe based FPGA accelerator cards. For applications with latency-critical CPU to FPGA interactions, such as computation kernel acceleration of scientific computing algorithms, the FPGA coprocessor approach can provide over 10 times lower latency.

However, the actual performance speedup can be severely limited by memory bandwidth and memory access latency. Since all memory accesses from the FSB-FPGA modules need to go through the FSB on the host computer, all five FPGA chips need to share the same FSB connection, and all four sockets on the same server motherboard share the same bus as well. For example, on a 4 socket Intel Xeon motherboard platform with 4 channels of DDR2-667 memories, a total of 21GB per second peak memory bandwidth is shared among all four CPU sockets. Hence each FSB-FPGA stack gets only 5.3GBps memory bandwidth, and each of the 5 user FPGAs in the stack has only 1GBps memory bandwidth available on average. In comparison, when a single DDR2-667 memory channel is directly connected to a FPGA chip, each FPGA gets the full 5.3GBps peak memory bandwidth.

In addition, all physical DRAM accesses from the FSB-compute modules need to go through the FSB-base module and host motherboard north-bridge chip. The additional overhead of inter-FPGA communication from FSB-compute to FSB-base modules and the FSB communication layer protocol can add a couple hundred nanoseconds of latency on top of the typical 40 nanosecond memory controller access time, which effectively quadruples system memory access time from FPGA. On the other hand, when communicating to host CPU memory cache contents, the FSB-FPGA approach can drastically shorten the latency. Since the FSB protocol enables cache coherent access of memory locations resident in each CPU's own cache, direct memory accesses between FSB-FPGA modules and memory which is resident in the host CPU cache can be directly transferred via FSB without going to system DRAM. This characteristic of the FSB-FPGA approach makes it perfect for accelerating small computational kernels within sequential codes executing on the host CPU with a small amount of shared data sets.

3.5 FPGA compute cluster

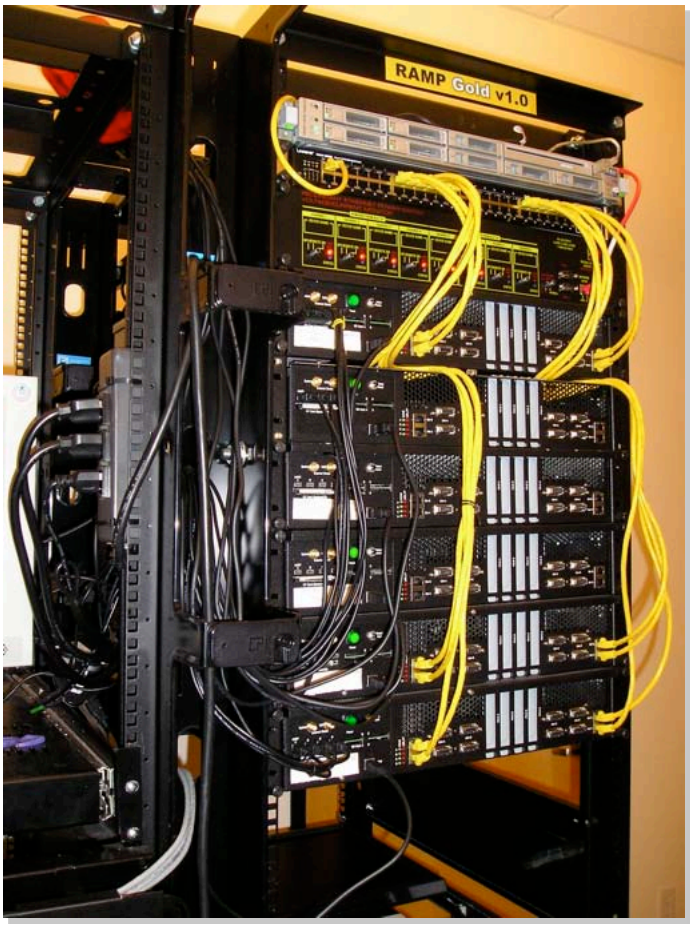

Figure 6: 6-node BEEcube BEE3 system at UC Berkeley

An FPGA compute cluster, such as the BEEcube BEE3 system [5], provides a scalable approach to achieve both large emulation capacity and easy software integration with a host computer.

Unlike the case of FPGA based emulation solutions, where the total capacity is limited to the number of FPGAs on the same or a few tightly integrated PCBs, FPGA compute clusters scale by interconnecting a large number of compute modules via commercial high bandwidth network interfaces, such as 10G-Base-CX Ethernet. Hence the total emulation capacity is scalable to several server racks worth of modules, on the order of 100 modules. With ample network bandwidth through either direct inter-module links or network switches, each FPGA compute cluster can be optimized for the specific application at hand by physically or logically reconfiguring the network topology and connectivity.

At the same time, each module can also connect to multiple front-end computer nodes via a built-in PCIe-over-cable solution to allow a flexible ratio between FPGA to front-end conventional compute nodes, ranging from 1 FPGA chip to 1 processor chip, all the way to a single front-end computer to an entire 100 node (400 FPGA) cluster. This solution not only overcomes the capacity limitation of FPGA accelerator cards, but also reduces the overall system cost by allowing the use of a large number of cheaper medium capacity FPGA chips rather than the very expensive highest capacity ones. At the high end of capacity, an FPGA with twice the capacity typically cost over four times the price.

From a memory bandwidth and latency perspective, FPGA compute clusters provide the most scalable aggregate memory bandwidth, while achieving the lowest DRAM latency at the same time. For example, each of the FPGA nodes on the BEE3 system has two independent DRAM channels, each at a DDR2-667 interface speed. A total of 10.7GB per second peak memory bandwidth is available for each of the FPGA in the system. With each DRAM controller directly implemented on the FPGA, the memory latency can be reduced down to just 40 nanoseconds for typical access patterns. Each BEE3 module contains 4 FPGAs, which adds up to 8 independent memory channels with an aggregate of 42.8 GB per second bandwidth and up to 64GB of physical memory capacity. A standard 42U 19-inch server rack can hold up to 21 BEE3 modules (2U each), which provides over 1.3TB of DRAM capacity and 0.9TB per second memory bandwidth.

From an interconnect bandwidth perspective, FPGA compute clusters also provide the most flexible network topology, as well as the most scalable network throughput. In the case of the BEE3 system, each FPGA node directly interfaces with two 10Gbps Ethernet CX4 ports. Each BEE3 module has eight 10Gbps ports, providing up to 160 gigabits per second full duplex network throughput. In the same 42U server rack with 21 BEE3 modules, the total aggregate network throughput exceeds 3 terabits per second. In addition to ample network bandwidth, the usage of standard 10 gigabit Ethernet interface and cabling infrastructure allows users to not only build packet switched networks with COTS (Common Off The Shelf) 10G Ethernet switches, but also custom low latency circuit switched networks just with the physical layer

multi-gigabit transceivers. For example, with eight 10G-Base-CX4 ports on each BEE3 module, users can form all network topologies ranging from a 1D/2D mesh/torus to a full 9-ary hypercube. When linked directly between BEE3 modules, the direct FPGA-to-FPGA network latency can be as low as just 150ns per hop inter-module, and 20ns intra-module.

4 Results and Discussion

To compare and contrast the various pros and cons of the above five classes of large scale multi-core processor emulation solutions, this study focuses on the analysis of host hardware differentiation, software design environment, and user support effort. At the end of this chapter, a real-world example of emulating a full 64-bit multi-core processor is included as a case study to detail the usage model of various prototyping approaches as well as the design trade-offs.

4.1 Host Hardware Differentiation

For each class of the hardware emulation platforms included in this study, the following subsections analyze their capacity, scalability, simulation speed, flexibility, accuracy, and cost. The table below summarizes the qualitative results of the various comparison criteria, where each individual subsection details the quantitative analysis of each metric.

Table 1: Emulation Systems Comparison Overview

	Scalability	Capacity	Speed	Flexibility	Accuracy	Cost
ASIC Gate-level Accelerator	C	C	C	C	A	>$1M
Multi-FPGA ASIC Emulator	C	C	B	B	A-	$100K-250K
FPGA Accelerator Card	B	C	B	B	B	<$10K
FPGA Coprocessor Module	B	B	A	A-	B	<$100K
FPGA Compute Cluster	A	A	A-	A	A-	<$100K

4.1.1 Capacity

System capacity is determined by the host system logic and memory capacity. Logic capacity is defined as the total number of logic gates available for the target processor designs, and memory capacity is defined as total main system memory in gigabytes available. Logic capacity has been the most common comparison metric among hardware emulation systems, mainly due to the lack of logic capacity in the early days of the hardware emulation era. Emulation vendors use logic capacity as a marketing tool for advertising their own system as either the largest logic capacity in emulated gate counts or the highest logic gates per dollar.

While the absolute logic capacity is critical for the emulation of large flat ASIC gate level netlists, for computer architecture exploration of design trade-offs in early to mid-stage processor development, logic capacity alone is no longer the king of emulation metrics. In various FPGA Architecture Model Execution (FAME) [8] schemes proposed by Research Accelerator for Multiple Processors (RAMP) community researchers [7], logic capacity can be traded-off with execution time. For example, in the multi-threaded scheme, such as the Berkeley RAMP Gold

implementation of the SPARC V8 Instruction Set Architecture (ISA), 64 processor cores are executed on a single multi-threaded target processor. The logic capacity of this implementation is about the same as a single processor core implemented in its RTL form, but the 64-way multi-threading allows execution of 64 processors at 1/64th of the single processor throughput. Nevertheless, each processor requires its dedicated memory space for their individual programs and data; hence memory cannot be easily virtualized or shared.

Given the importance of multi-threaded schemes in early stage processor architecture research, next generation processor emulation platforms must provide hundreds of gigabytes to terabytes of system memory capacity at an affordable cost. Both ASIC Gate-level Accelerators and Multi-FPGA ASIC Emulators are optimized to provide the maximum logic capacity in an integrated system for ASIC gate-level verifications, where memory requirements have traditionally been limited to relatively small amounts of Static Random Access Memory (SRAM) that can fit onto the final IC implementation. Most of these types of systems have only tens or hundreds of MB of external SRAM, with some capable of extending the memory capacity with custom add-on memory modules, both of which are clumsy and expensive. FPGA accelerator cards can use commodity Dynamic Random Access Memory (DRAM) Dual Inline Memory Modules (DIMMs), but its physical form factor limits the number of DIMMs to just a few gigabytes per card.

FPGA Coprocessor Modules can share the same main memory with the host computer processor, and therefore have the potential to access the full system memory capacity of the host server. However, the memory accesses from the FPGA coprocessor need to be carried through the system north-bridge device, which adds latency and complicates the FPGA firmware design drastically. FPGA compute clusters, on the other hand, use the same commodity server DRAM DIMMs, but directly connect them to each FPGA on multiple memory channels with multiple DIMMs per channel. For example, each BEE3 system consists of 4 FPGA chips, each with 2 DRAM channels and 2 DIMMs per channel. This approach allows the maximum amount of DRAM capacity and throughout available at the disposal of FPGAs with minimal memory latency.

4.1.2 Scalability

Scalability of emulation platforms is defined as the range of capacity expansion (both logic and memory) from a minimum system configuration to its maximum. In the cases of ASIC Gate-level Accelerators and Multi-FPGA ASIC emulators, the range of capacity scaling is limited by the combination of the relatively large logic capacity of the minimal configuration and the few available steps to reach their maximum capacity. Low latency inter-chip connectivity approaches found on these two types of systems severely limit the number of integrated minimum size modules to just a few.

FPGA Accelerator Cards and FPGA Coprocessors are both tied to the host computer node on the system bus, which usually limits the inter-FPGA communication mode to take place through the host computer network layers. Although this scheme can be useful for computational acceleration of scientific applications using message passing communication schemes, for tightly coupled processor architecture modeling applications, this approach limits the capacity of the system to that which can be fitted into a single host computer server node.

FPGA compute cluster approaches address the scalability issue by directly integrating high bandwidth low latency links on the FPGA processing boards. In the case of BEE3, each module consists of 8 CX4 10Gbps full duplex network interfaces, which can be configured into many interconnect topologies, ranging from simple 3D mesh to 8-dimensional hypercube. The only scaling limitation comes from the physical interconnect length. At up to 50 ft of cable length on each CX4 interface, up to 64 racks worth of 1024 BEE3 modules could be integrated in the same system, providing up to 5 billion logic gates of capacity, 64 terabytes of system DRAM, and 160 Tbps interconnect bandwidth. This magnitude of scalability is unrivaled by any other existing platform.

4.1.3 Speed

Speed is measured in terms of target processor cycle executions per second. With the exception of ASIC Gate-level Accelerators, which have a maximum emulation clock rate of just a few MHz, the other four types of FPGA based emulation platforms all have the potential to run target processors at a clock rate ranging from a few tens of MHz to a couple hundreds of MHz. Among the FPGA based systems, system-level host architecture features can have a profound effect when simulating complex multi-core processors using various simulation acceleration schemes. A target processor architecture model can be partitioned into multiple physical FPGAs across multiple physical modules, and the behavior of the processor can also be partitioned into functional models running on conventional general purpose CPUs and timing models running on the FPGA systems. These two types of partitioning problems create two potential bottleneck in the host platform. One is the interconnect latency and bandwidth between target processor cores, and the other is CPU to FPGA latency and bandwidth between functional and timing models. Another kind of bottleneck rises from memory access latency and bandwidth between the target processor core and the FPGA platform's memory subsystems.

Each of these three potential bottlenecks can be the dominating speed reduction factor for a given processor modeling technique and target processor size and complexity. When using simple RISC processor cores, many of them can be implemented in a single FPGA chip, but the memory bandwidth requirement grows linearly with the number of processor cores running simultaneously. 32 cores running at 100MHz each with a 32bit architecture at 25% load/store rate will require a minimum of 3.2GBps sustained memory bandwidth, which requires at least

two physical DRAM channels running at Double Data Rate 400MHz (DDR400) rates or above. On the other hand, a large super-scalar processor core might take the entire FPGA logic capacity, hence the memory bandwidth would no longer be the bottleneck; however, the inter-FPGA communication latency can directly limit the yielded speed of the target architecture. In the case of hybrid functional/timing modeling techniques, where the functional model runs on a conventional CPU, the interaction latency between the functional and timing models is critical for the speed of the overall simulation.

4.1.4 Flexibiliy

Flexibility of FPGA emulation platforms is measured with respect to how many different architecture-modeling techniques each host system can support. For the purpose of this study, we focus on hardware based processor architecture modeling approaches, instead of software simulation based approaches. A number of different emulation strategies and architecture modeling techniques have emerged from the research community [7], including direct RTL, decoupled RTL, abstract models, multi-threaded models, and hybrid models.

Direct RTL is the most universal modeling technique used by all traditional ASIC emulators. A full ASIC gate-level netlist or RTL source code is required to allow the host emulation system to produce the exact cycle-to-cycle functionality of the final silicon implementation in the host emulation system. In most cases, synthesizing an ASIC optimized netlist or RTL to a FPGA target is such an inefficient mapping process that the host FPGA implementation of the ASIC gate-level netlist can be up to five time larger than the FPGA optimized implementation with the exact same cycle-to-cycle functionality. Due to this large logic capacity requirement (typically over 5 million gates) for full ASIC gate-level emulation, the direct RTL approach is not well suited for FPGA accelerator cards or FPGA coprocessor modules. Dedicated ASIC Gate-level Accelerators or Multi-FPGA ASIC emulators are specially designed to hand the direct RTL method, and FPGA compute cluster can be scaled to achieve this large logic capacity as well.

Decoupled RTL removes the restriction of a pure ASIC gate-level netlist or RTL, and instead allows the use of an FPGA optimized implementation to emulate the intended functionality. Furthermore, host system clock cycles can be decoupled from the target design cycle. For example, to emulate a multi-port register file with 3 write ports and 5 read ports at a desired 100MHz target clock rate, a single FPGA optimized dual-port BRAM running at 400MHz host clock rate can be used to model the same functionality of a single target clock cycle of the multi-port register file with four host clock cycles. The input and output data are time multiplexed onto the host optimized BRAM implementation. By decoupling the FPGA implementation from the original ASIC gate-level implementation, the mapping efficiency can improve by a factor of 2 to 5 times that of the direct RTL approach. Therefore all FPGA based hardware emulation systems can support the decoupled RTL method.

Abstract models allow behavior modeling techniques to be used in place of RTL implementation models. In particular, separate ISA functional models and processor timing models can be combined with no direct correspondence to their final structural implementation in silicon. This approach allows the evaluation of several different ISA models decoupled from their pipeline timing models, as well as the same ISA model to be shared with many different implementation timing models. With the two models separately, part of or the entire ISA functional model can be implemented on a host CPU based system tightly coupled with the FPGA host. For example, an ISA model of the Intel x86 instruction set is best implemented natively on a host Intel CPU, where the pipeline timing model is implemented on an FPGA emulation system attached directly to the CPU system bus or PCI bus. Given the requirement of host CPU integration, only an FPGA host system with a direct connection to the host CPU can support this abstract model.

A multi-threaded processor model further time multiplexes its functional and timing model pipeline among many different instances of processor cores of the same type. Instead of stamping out multiple identical processor cores on the host FPGA system, the multi-threaded approach uses a single multi-threaded processor model to emulate the function of many cores, while keeping each processor's internal state separately. The multi-threaded processor model only slightly enlarges the physical logic footprint of the emulation by adding the data multiplexors, but the memory capacity requirement grows linearly with the number of cores emulated. A multi-threaded processor core model can easily handle 16 cores with its logic implementation, but will require 16 times the system memory capacity to keep all states separately. Therefore only the host emulation system with ample system memory capacity can take full advantage of the multi-threaded method.

In practice, users combine all of the above four modeling techniques to achieve their design goals, which results in what is called a hybrid model. For example, a user may start with highly abstract models of the processor, decoupled function and timing models, and multi-threaded execution to analyze the system behavior. Later on in the design cycle, the user could then refine the various models to be closer to their final RTL implementation. Finally, when all the models have been implemented, a full direct RTL model can be used to validate the final results.

These techniques, along with the primary types of FPGA emulation platforms, are summarized in table the below. The body of the table illustrates which platform type is appropriate for which emulation strategy.

Table 2: Emulation System Flexibility Comparison

	Direct RTL	Decoupled	Abstract	Multi-threaded	Hybrid
ASIC Gate-level Accelerator	X				
Multi-FPGA ASIC Emulator	X	X			
FPGA Accelerator Card		X	X	X	
FPGA coprocessor module		X	X	X	
FPGA compute cluster	X	X	X	X	X

4.2 Software Design Environment

Software design environments for processor architecture development can be grouped into three main approaches: ASIC gate-level translation, FPGA optimized synthesis, and processor model abstractions.

4.2.1 ASIC gate-level translation

For traditional ASIC emulation systems, either proprietary systems like Cadence Palladium or open FPGA based systems like Synplicity HAPS-54, the firmware and software requirements are restricted to simply mapping ASIC gate-level netlists to that of the target emulation hardware. In the case of dedicated ASIC gate-level accelerators, proprietary tools are provided by system vendors to directly map the ASIC gates to machine code for their specialized gate-level processors in the host emulation system.

FPGA based ASIC emulators mainly rely on the FPGA backend implementation tool flow provided by FPGA vendors to synthesize basic ASIC gate-level netlists into FPGA configuration bit files. The main challenges come from how to partition a large flat ASIC gate-level netlist across multiple target implementation chips (i.e. FPGAs) without being limited by the cross chip I/O bandwidth and latency. This is the main reason traditional ASIC emulators are limited to just a few MHz of emulated clock rate for the final ASIC gate-level design. Despite continuous efforts from the Electronic Design Automation (EDA) industry to try to simplify the partition problem, with tools such as Synplicity Cerify, the bulk of the engineering effort is still on determining how to map the flat ASIC netlist onto multiple FPGAs, hence leading to common use of the largest capacity FPGA chips to reduce the interchip communications requirements. With top-of-the-line FPGA chips, such systems also cost exponentially higher than other alternative solutions.

Another challenge comes from the gate-level translation of ASIC technology to FPGA technology. For example, an efficient ASIC implementation of a multi-port register file (e.g. two write ports, three read ports) does not have an efficient direct target implementation on the FPGA. If one simply synthesizes the original ASIC gate-level netlist with the FPGA backend synthesis flow, such a register file will be translated into a large collection of multi-port register bits, which consume a large volume of FPGA logic resources. An alternate strategy is to

functionally emulate the register file with FPGA-efficient resources, such as dual port Block RAMs (BRAMs). Multiple read/write ports can be emulated by using time multiplexes accesses to the physical dual-port BRAM. Such transformations from an ASIC gate-level netlist to a FPGA target currently require manual engineer effort to create the efficient FPGA implementation and verify the functionality.

4.2.2 FPGA optimized synthesis

Instead of directly mapping ASIC gate-level netlists, the FPGA optimized synthesis approach provides subsystem functional-level equivalence while utilizing the FPGA resources efficiently. Although the FPGA implementation does not match the ASIC gates exactly, its functionality is identical down to the exact clock cycles and bit accuracy. This approach typically leads to a 3X to 5X reduction of FPGA resources required for emulating the same system with minimum impact on the accuracy of the emulation.

Users typically start with a generic RTL description of the system and parameterize those components that are different between ASIC and FPGA, such as memory blocks. Even though the different implementations of these library components need to be validated to match timing and functionality between ASIC and FPGA, most of the designs can be synthesized to the corresponding technology target with identical cycle-to-cycle functionality. This is the prevailing method for commercial processor core vendors to provide a cycle accurate emulation of their processor cores running at a sufficiently high clock rate (i.e. 50MHz) on a single or multiple FPGAs, especially for processor cores that can be tailored to the specific applications domain of interest, such as the Tensilica processor cores [10].

4.2.3 Processor model abstraction

A key disadvantage of the two software design environments in the previous subsections is the low user productivity in creating new designs. Both ASIC gate-level translation and FPGA optimized synthesis design approaches rely on the existing ASIC/FPGA synthesis driven CAD flows. These CAD tools are designed to achieve the best silicon performance in terms of clock rate and/or resource utilization, at the expense of user design efforts. The abstraction level of RTL source code for FPGA or ASIC is so low that the user is responsible for all clocking, control, and resource management. Existing hardware description languages, such as VHDL or Verilog, force users to implement so many hardware details that it is commonly overwhelming for new users, especially for those computer architects who are only interested in modeling the processor rather than fully implementing it.

Processor model abstraction approaches the design problem from a high level modeling perspective, through a combination of high-level design language usage and a library of parameterized components. For users who are only interested in speeding up the modeling of a

processor's architecture, the absolute performance of their FPGA implementation is not critical, but rather the turn-around time of design modifications. Using high-level hardware design languages, such as Bluespec System Verilog [9], users can improve their productivity by a factor of five over existing HDL approaches. In terms of the absolute implementation performance results generated by a high-level design language may be a factor of two slower or use 25% more resources than the equivalent hand coded HDL results. But for early stage processor architecture design, a slightly slower FPGA based emulation is still four orders of magnitude faster than software simulation approaches.

4.3 User Support Efforts

Given the complexity of FPGA emulation systems, end user support for design creation and debugging is essential to the productivity of the designers. Due to the flexibility of the hardware, emulation systems are inherently more difficult to predict all of the possible user issues, which can range from simple functional bugs, timing closure issues, to more elusive "Heisenbugs". Especially for traditional FPGA based ASIC emulator systems, when a particular issue occurs, the first question is usually whether it's a problem with the user design or the underlying FPGA. The majority of these issues are related to high speed interfaces, clock domains, and timing issues. For system architects who prefer to explore the design space rapidly, these low-level issues can drastically reduce their overall productivity.

Users of BEE3 systems are presented with a virtual design sand box that shields them from the harsh physical design environment of using raw FPGA resources. All external interfaces, such as high speed DRAM, networks, inter-chip communication, analog-to-digital interfaces, host computer communications, have been abstracted into simple First In First Out (FIFO) based data sources/sinks with simple control handshaking protocols. Not only are the hardware interfaces and timing closure automatically generated by the BEEcube Platform Studio design environment, but so are all the initialization and software device settings for these interfaces. Hence the designers can concentrate on their core architecture design, rather than the implementation details of the underlying FPGA chips.

Another important aspect of end usability is the system scalability and manageability. For example, when using a single FPGA system, direct communication through Joint Test Action Group (JTAG) or serial ports are the most convenient methods of host connectivity. However, for large scale systems, like the BEE3, which has four FPGAs per module and up to 20 modules per rack, communication to 80 FPGAs via JTAG or a serial port is no longer an option. In these situations, gigabit Ethernet is currently the most cost effective method for host control of large scale emulation systems, while high-bandwidth inter-module data requires multiple direct 10Gbps links. This approach turns the original FPGA emulation system into a large scale compute cluster, where multiple networks are present for various functions of the system.

Additional host management software and cluster management middleware is required to present a large scale system as a single entity to the end user, or to allow multiple users to share a large installation without interference from each other. All these capabilities are enabled by removing the end user from the physical I/O interface directly on the FPGA system.

4.4 Case Study: OpenSPARC on BEE3

This section describes a case study using the BEE3 FPGA emulation platform for a number of different processor modeling techniques. The processor of choice is the OpenSPARC T1 processor from Sun Microsystems, which is the open source version of the UltraSPARC T1 processor. It is a highly integrated processor that implements the 64-bit SPARC V9 architecture. This processor targets commercial applications such as application servers and database servers. The features of the OpenSPARC T1 processor include:

- 8 SPARC V9 CPU cores, with 4 threads per core, for a total of 32 threads
- 132 Gbytes/sec crossbar interconnect for on-chip communication
- 16 Kbytes of primary (Level 1) instruction cache per CPU core
- 8 Kbytes of primary (Level 1) data cache per CPU core
- 3 Mbytes of secondary (Level 2) cache – 4 way banked, 12 way associative shared by all CPU cores
- 4 DDR-II DRAM controllers – 144-bit interface per channel, 25 GBytes/sec peak total bandwidth
- IEEE 754 compliant floating-point unit (FPU), shared by all CPU cores

Sun's UltraSPARC T1 processor has been designed to incorporate hypervisor technology in order to present a virtualized machine environment to any guest operating system running upon it. The resulting software model for a guest operating system is referred to as the "sun4v" architecture. This virtual machine environment is implemented with a thin layer of firmware software (the "UltraSPARC Hypervisor") coupled with hardware extensions providing protection. The UltraSPARC Hypervisor not only provides system services required by the operating system, but it also enables the separation of platform resources into self-contained partitions (logical domains), each capable of supporting an independent operating system image.

The goal of mapping the OpenSPARC core to FPGA emulation targets is to provide an emulation environment for experimenting with multiple OpenSPARC cores along with nearby application domain specific accelerators on the target System-on-Chip (SoC) solution. Given the complexity of the processor, direct RTL mapping of the exact ASIC gate-level implementation of just a single core requires over 500 thousand logic slices spread over a couple of the largest capacity Xilinx Virtex5 FPGA chips.

In order to achieve a cost effective approach to this modeling problem, a two-fold solution was used to reduce the overall system cost. One half of the solution is to use a scalable FPGA compute cluster, the BEE3 system, as the host emulation platform to provide the balance between overall system cost and hardware partitioning complexity. The other is to reduce the logic capacity requirement by optimizing the OpenSPARC model for architecture exploration rather than gate-level validation.

The logic reduction process started with decoupled RTL of key processor components with FPGA optimized implementations that use just a fraction of the resources. Register files are translated into multi-cycle BRAM accesses, which combine all register bits into memory bits in FPGA efficient block RAM components. Furthermore, abstract models are created to replace complex memory and network connections, such as the full on-chip crossbar and multi-level memory hierarchy.

In order to provide emulation of full-scale processors with full binary software compatibility, the target emulation system has to provide complete memory and I/O interfaces required for the full software operating system and applications. Directly mapping the exact silicon RTL implementation of the full multi-level cache, reverse directory lookup, crossbar links, and network I/O interfaces, consumes too many FPGA logic resources to fit a single core in each FPGA.

Instead of the full RTL implementation, a single Microblaze processor core is used to run all software needed to emulate the full OpenSPARC memory and I/O subsystems via the master FPGA. All slave processor cores communicate to the master core for memory and I/O access to the physical DRAM and external network interfaces. Although the performance of this software memory emulation layer is suboptimal, it provides an easy debugging environment to achieve the full binary software compatibility of running OpenSPARC cores with existing Solaris operating system and applications.

In addition, the software emulation abstraction drastically reduced the logic resource requirements. Each BEE3 module can emulate up to 4 OpenSPARC T1 cores, one in each of the Xilinx Virtex5 LX155T FPGA chips. From the original ASIC gate-level equivalent of over 500 thousand logic slices, the abstract model of OpenSPARC T1 processor core with identical functional and timing behavior consumes just 100K logic slices—a quarter of the original side. The leftover resources on each FPGA are enough to accommodate additional accelerators and I/O interfaces. The BEE3 emulated OpenSPARC T1 cores runs at 66MHz, and is capable of booting a full Solaris operating system with binary compatibility to compiled versions for standard SPARC computers.

5 Conclusions

Accurate and rapid modeling of large-scale multi-core processors is crucial to achieve the requirements of next generation high performance computing system development. Traditional software simulation based approaches to processor architecture exploration is no longer adequate in the era of multi-core and many-core processor architectures. The requirement to execute full user applications with operating system support has pushed computer architects and software developers to take advantage of hardware based emulation systems, in order to achieve over 2 orders of magnitude speedup versus software simulation approaches.

However, the wide selection of existing commercial hardware emulation systems in combination with the various processor modeling methods have presented a convoluted and confusing decision space for potential computer architectures. This study makes an attempt to clarify this selection space by classifying the hardware emulation systems into five types, and comparing their effectiveness when targeting four different processor architecture execution models. Although each type of hardware emulation system can achieve the best result in a single category of comparison, only the FPGA compute cluster approach achieves the best or second best in all categories of measurements.

For design teams architecting many-core processors from the ground up, it is crucial to select the most versatile and scalable emulation system to allow designers to move from one processor architecture execution model to another throughout the design process. Different design teams must have the option to start with different architecture execution models, based on their existing approaches, design concept maturity, and exploration space. A team may start with a highly abstracted or multi-threaded architecture emulation approach to allow flexibility in subsystem modeling and integration, but later move onto a direct RTL implementation as subsystem design details are committed to implementations.

6 Recommendations

From the results in the previous sections, we recommend the FPGA compute cluster as the most versatile processor architecture prototyping platform to accommodate the widest range of modeling techniques with the best or near best of all performance metrics, balancing cost and scalability. Each processor design team will first need a medium scale prototype system to experiment on a single socket design with tens to a couple hundred of cores. Once the single socket processor architecture design matures, the design team needs to move onto a large-scale prototyping system, where many processors can run simultaneously with near real-time software execution capability. The medium scale prototype system can be hosted locally to each design team to minimize design turn-around time, where the large scale system can be hosted remotely in a central location and shared by many design teams.

User support for advanced processor architecture development is very different from conventional logic emulation for ASIC chips. End-user support from prototyping system vendors needs to match user requirements for rapid design turn-around time, and system optimization as to best leverage the underlying FPGA resources. A productive user design environment requires not only smart tools to integrate user specific design intelligently with the host system platform, but also a rich library of host system infrastructure, such as memory hierarchy, interconnect abstractions, system I/Os, and debugging interfaces. End-users need to treat the host prototyping system as a computing platform, not just a blank logic fabric.

7 REFERENCES

[1] Cadence. Incisive Palladium III. http://www.cadence.com/products/functional ver/accel emul/pseries.aspx.

[2] Synplicity, HAPS High-performance ASIC Prototyping System. http://www.synplicity.com/products/haps/

[3] HiTech Global HTG-V5-PCIE board http://www.hitechglobal.com/Boards/V5PCIExpress.htm

[4] Nallatech Intel Xeon FSB FPGA Accelerator Module http://www.nallatech.com/?node_id=1.2.2&id=83&request=2008update

[5] BEEcube BEE3 FPGA Computer Cluster http://www.beecube.com/platform.html

[6] C. Chang, J. Wawrzynek, and R. W. Brodersen. **BEE2: A High-End Recongurable Computing System.** *IEEE Design and Test of Computers*, 22(2):114–125, Mar/Apr 2005.

[7] J. Wawrzynek, D. Patterson, M. Oskin, S.-L. Lu, C. E. Kozyrakis, J. C. Hoe, D. Chiou, and K. Asanovic. **RAMP Research Accelerator for Multiple Processors.** *IEEE Micro*, 27(2):46-57, 2007.

[8] DARPA Final Project Report, **FPGA Emulation for Computer Architecture Research and Development**, John Wawrzynek and Krste Asanovic, *in preparation*.

[9] Bluespec System Verilog, http://www.bluespec.com

[10] http://www.tencilica.com/

List of Acronyms, Abbreviations, and Symbols

Acronym	*Description*
ASIC	Application Specific Integrated Circuits
BRAM	Block RAM
CPU	Central Processing Unit
DDR400	Double Data Rate 400MHz
DIMM	Dual Inline Memory Module
DRAM	Dynamic Random Access Memory
EDA	Electronic Design Automation
FPGA	Field Programmable Gate Array
FIFO	First In First Out
FPU	Floating-Point Unit
FAME	FPGA Architecture Model Execution
FSB	Front Side Bus
IC	Integrated Circuit
ISA	Instruction Set Architecture
JTAG	Joint Test Action Group
PCI	Peripheral Component Interconnect
PCB	Printed Circuit Board
RTL	Register Transfer Level
RAMP	Research Accelerator for Multiple Processors
SMP	Shared-Memory multi-Processor
SRAM	Static Random Access Memory
SoC	System-on-Chip